Combinatorial optimization
CoContest semester project assignment: Air Tickets TSP

Industrial Informatics Research Center
http://industrialinformatics.fel.cvut.cz/

April 24, 2018

Abstract

This document introduces the assignment for the CoContest semester project.

1 Motivational example

The members of Industrial Informatics Research Center have submitted three conference papers to different conferences around the world. All their papers were accepted for the presentation, but the problem is that the conferences days are overlapping. Luckily, each of the conferences is scheduled for multiple days.

One of the paper authors, Theodor, was picked to present all the papers. However, it emerged that grant agencies' funds are being abused for so-called conferential tourism (one such popular destination has become the area of Eastern Asia, where cheap manpower is known to be offering unprecedented services to European tourists). This observation triggered harsh austerity measures that the funding agencies imposed on scientific workers. Therefore, Theodor needs to think carefully, which flights tickets he should buy such that he visits all the conferences and presents the papers.

Theodor needs to start at the city he lives in and each day, make exactly one flight to another destination. Finally, the last day, he returns to his hometown. Finding the cheapest flight sequence sounds like the problem he is dealing with in one of his conference paper entitled *Sub-exponential algorithm for solving the famous Traveling Salesman Problem*. However, some of the flights between specific destinations are not available during each day, and Theodor is puzzled, whether he can still use his algorithm to find the cheapest sequence of flights[1].

2 Air Tickets TSP - formal statement

Let $C = \{c_1, c_2, \ldots, c_n\}$ be a set of n cities and $c_s \in C$ be the city where the trip starts. Let $D = \{0, \ldots, n-1\}$ be a set of days. Let $F \subseteq C \times C \times D$ be a set of all available flights. Notice that some triples (c_i, c_j, d) might not be in F, since it is a subset of $C \times C \times D$. Each triple $(c_i, c_j, d) \in F$ denotes a flight ticket $c_i \in C$ to $c_j \in C$ on day $d \in D$. Its price is $p_{c_i,c_j,d} \in \mathbb{N}_0$.

Let $\pi = (\pi(1), \pi(2), \ldots, \pi(n))$ be a permutation of C. The goal is to find the trip π^* such that:

[1] This problem is inspired by a recent challenge given by Kiwi.com https://github.com/kiwicom/travelling-salesman

$$\pi^* = \arg\min_{\pi} P(\pi) \tag{1}$$

such that

$$\pi(1) = c_s \tag{2}$$

$$P(\pi) = p_{\pi(n),\pi(1),n-1} + \sum_{d \in D \setminus \{0\}} p_{\pi(d),\pi(d+1),d-1} \tag{3}$$

$$(\pi(d), \pi(d+1), d-1) \in F \quad \forall d \in D \setminus \{0\} \tag{4}$$

$$(\pi(n), \pi(1), n-1) \in F \tag{5}$$

$$\pi \text{ is a permutation of } C \tag{6}$$

In other words, you are asked to find the shortest Hamiltonian cycle on a time-dependent graph $C \times C \times D$ (where some edges might be missing), starting at c_s on day 0, finishing at c_s on day $n-1$. Hence, during the whole trip, you make $|C| = n$ flights, exactly one each day and you can consider that each flight transfers you immediately, i.e. traveling from city c_i to c_j on day d gives you access to all flights from c_j on day $d+1$.

3 Rules

If you decide to choose the contest as your semestral project, then you are expected to implement a correct solver for the Air Tickets TSP. The implementation will be submitted to BRUTE https://cw.felk.cvut.cz/brute/ where it will be automatically evaluated (number of submissions is not limited). The grading is combination of ability of finding good solutions and the achieved rank relative to other students (w.r.t. the objective function). Therefore, you can acquire some minimum number of points even if your solver is not very efficient relative to other students.

In BRUTE, you will find 3 tasks related to the contest. Each task has specific instances, rules and grading. The contest is split into different tasks so that we avoid re-evaluation of the instances (which is time-consuming) and so that you can implement specific solver for each task.

1. `SP_CC_O`: you have to implement an exact, MILP solver for the problem. If your solver solves optimally all the instances in this task, then you will get 3 points for this task. If the solver returns suboptimal solution for **any** instance in this task, then the evaluation of your solver is stopped and you will get 0 points in this task.

2. `SP_CC_T`: the goal is to find the best possible feasible solution within the specified time limit, i.e. the optimal solutions are not required and you are encouraged to implement clever heuristics solving these instances. For each instance in this task, you will obtain one point if the length of the trip in your solution is not worse than our threshold (5 points at max).

3. `SP_CC_R`: similarly as in `SP_CC_T`, in this task we are also interested in finding the best possible feasible solution within the specified time limit. However, the evaluation of your solver will depend on how good your solver is relative to other students' solvers, i.e. the number of points obtained will depend on your rank (3 points at max).

Some general contest rules also apply

1. usage of single-purpose problem-specific solvers is prohibited (i.e. a MILP solver is allowed, but somebody's else code for solving the Air Tickets TSP is not).

2. every participant is required to write its own code. However, sharing ideas and other discussion about the problem is encouraged

4 Input and Output Format

In `SP_CC_O`, your solver will be called as

```
$ ./your-solver PATH_INPUT_FILE PATH_OUTPUT_FILE
```

whereas in `SP_CC_T` and `SP_CC_R` we include a time-limit

```
$ ./your-solver PATH_INPUT_FILE PATH_OUTPUT_FILE TIME_LIMIT
```

- `PATH_INPUT_FILE` and `PATH_OUTPUT_FILE`: similarly as in homeworks, these parameters represent the path to the input and output files, respectively (see below for description of the file formats).
- `TIME_LIMIT`: an integer representing the time-limit in seconds given to your solver. Your solver will be killed after the time-limit is reached and the solution written in the output file (if any) will be taken as the result of your program.

The input file has the following form (we use one space as a separator between values on one line)

c_s
$c_1 \quad c_2 \quad 0 \quad p_{c_1,c_2,0}$
$c_1 \quad c_2 \quad 1 \quad p_{c_1,c_2,1}$
$\vdots \quad \vdots \quad \vdots \quad \vdots$
$c_1 \quad c_2 \quad n-1 \quad p_{c_1,c_2,n-1}$
$c_2 \quad c_1 \quad 0 \quad p_{c_2,c_1,0}$
$c_2 \quad c_1 \quad 1 \quad p_{c_2,c_1,1}$
$\vdots \quad \vdots \quad \vdots \quad \vdots$
$c_2 \quad c_1 \quad n-1 \quad p_{c_2,c_1,n-1}$
$\vdots \quad \vdots \quad \vdots \quad \vdots$
$c_n \quad c_{n-1} \quad n-1 \quad p_{c_n,c_{n-1},n-1}$

where c_s and c_i's are three characters codes of the airport (e.g. `PRG`), $d \in D$ is the day at which is a ticket available for price $p_{c_i,c_j,d} \in \mathbb{N}_0$. c_s denotes the airport where the trip has to start and finish. Please notice that some triples (c_i, c_j, d) may be missing, as there is no flight between c_i and c_j on day d.

The output file consists of two lines

$P(\pi)$
$\pi(1) \quad \pi(2) \quad \cdots \quad \pi(n)$

where π is a feasible permutation of cities and $P(\pi) \in \mathbb{N}_0$ is the total cost of trip π. Notice that as we need to start the trip from city c_s, then always $\pi(1) = c_s$ holds.

In theory, there might be no feasible solution. However, we ensure that for every instance, at least one feasible solution exists.

Example 1

Input:

```
COW
ADZ COW 0 906
ADZ COW 1 984
ADZ COW 2 443
ADZ LST 0 1474
ADZ LST 1 978
ADZ LST 2 560
```

```
COW ADZ 0 29
COW ADZ 1 188
COW ADZ 2 111
COW LST 0 58
COW LST 1 1051
COW LST 2 82
LST ADZ 0 753
LST ADZ 1 778
LST COW 1 1367
LST COW 2 1291
```

Output:

```
1279
COW LST ADZ
```

The optimal trip is given as

$$\texttt{COW} \xrightarrow{58} \texttt{LST} \xrightarrow{778} \texttt{ADZ} \xrightarrow{443} \texttt{COW}$$